1 Introduction

The results shown in this report are obtained from running LDAS at 1/4° using 30 minute and 1 hr timesteps. The output is written in binary format. GEOS base forcing is used in the runs. No observed forcings are considered for these results.

From the results obtained during baselining, using NOAH at 1/4° on the SGI AMES systems, the computational intensity was measured to be approximately 4.4-5 ms/gridcell/day.

2 Code Improvements

It was noticed that the subroutines that takes vegetation greeness fraction and surface albedo data and interpolates to the actual values for a certain date was being invoked during every timestep. These routines involve reading of greeness and albedo files for interpolation and was modified to be read only once a day.

The land surface processes have rather weak horizontal coupling and can be parallelized. For NOAH, the domain was decomposed statically based on the number of processors, with each processor working on an equal chunk of the domain. We performed experiments with the pool of tasks design, where the computational load is automatically balanced. In those experiments, the number of tiles calculated by each processor was identical, leading to the conclusion that computationally NOAH LSM runs are not significantly different for different parts of the domain.

The land surface model runs are performed in parallel by each processor in their own domain. A master processor is designed to handle the preprocessing before land surface model runs, and the output. The communication overhead in this model of computation is minimal since the processors communicate with the master only for decomposing the domain and for gathering information before writing output.

The temporal interpolation routines were improved by eliminating the ocean points. The spatial interpolation is conducted by calling the library routine called `ipolates`. The source code for ipolates were optimized for bilinear and budget options to improve the performance.

The cleanup and optimizations performed in the code also includes the use of internal files in fortran to eliminate unnecessary I/O in the program. Some of the direct access file routines were converted to sequential access routines also.

3 Results

Figure 1 shows the improvements in the code and the corresponding runtimes. It can be seen that for 30 minute and 1hr timesteps, the best performance of the code so far is 1.3 and 0.99 ms/gridcell/day respectively.

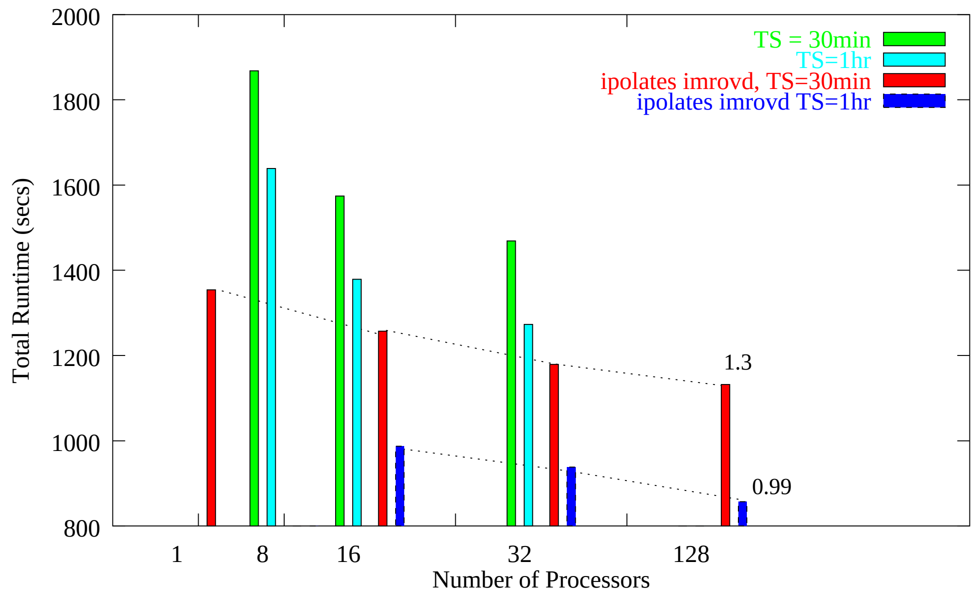

Figure 1: NOAH execution times on HALEM

The current code is further analyzed to identify the execution times of the individual components. Figure 2 shows the cumulative effect of different components on the overall performance of the code. The numbers are calculated for a respresentative run using 16 processors and 30 minute timesteps. The computational intensity numbers are also shown alongwith each component.

It can be seen that in the current code, the interpolation routines contribute significantly to the computational intensity of the code. The sequential execution of the code requires a computational intensity of 1.6. Since NOAH LSM is a very fine grained model, performance improvement by parallelization is not very significant. The contributions of each component as percentages of the total execution time is shown in Table 1.

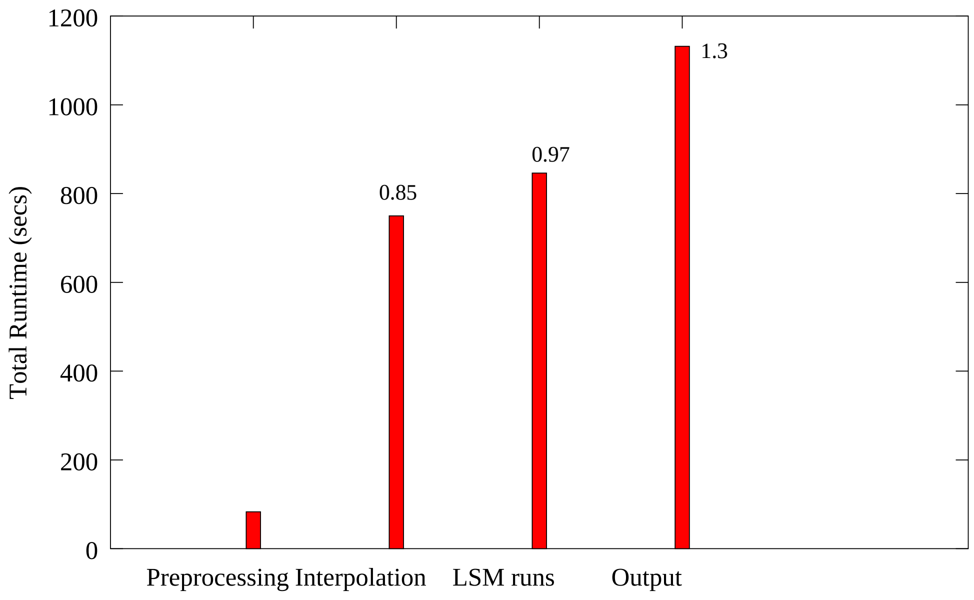

Figure 2: Componentized NOAH execution times

Table 1: Percentage of total computational time for different components

Routine	Percentage
Preprocessing	7.3
Interpolation	58.0
ipolates	46.0
zterp	12.0
LSM runs	8.5
Output	25.0